

AD-A049 849

STANFORD UNIV CALIF DIGITAL SYSTEMS LAB
EFFECTS OF FAILURES ON PERFORMANCE OF GRACEFULLY DEGRADABLE SYS--ETC(U)
DEC 76 J LOSG
DSL-TN-103

F/G 9/2

N00014-75-C-0601

NL

UNCLASSIFIED

| OF |

AD
A049 849



END
DATE
FILMED
3 - 78
DDC

AD A 049849

Center for
Reliable
Computing

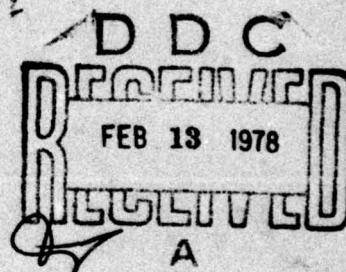
12

EFFECTS OF FAILURES
ON PERFORMANCE
OF GRACEFULLY DEGRADABLE SYSTEMS

Jacques Losq

Technical Note No. 103

December 1976



Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California

This work was supported in part by the National Science Foundation under Grant No. MCS 76-05327; and in part by the Joint Services Electronics Program (JSEP) under Contract No. N00014-75-C-0601.

(See 1473)

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

EFFECTS OF FAILURES
ON PERFORMANCE
OF GRACEFULLY DEGRADABLE SYSTEMS

Jacques Losq

Digital Systems Laboratory
Stanford University
Stanford, California 94305

ABSTRACT

The recent development of multiprocessor systems that offer resistance to faults by gracefully degrading after a failure opens vast new ranges of applications for fault tolerance and high reliability. The paper presents a general model for the evaluation of such systems. It takes into account the internal structure of the hardware, the characteristics of the various detection mechanisms, the unreliability of the software and even the type of applications these systems are used for. It provides many measures of the systems' performance such as: availability, meantime between crashes, average processing power and proportion of time spent in degraded mode. System optimization gives the best values for the number of processors, memories, ..., and shows the trade-offs between hardware and software fault-detection mechanisms. The model is illustrated by a concrete example.

Index Terms: Digital systems, fault-tolerance, graceful degradation, availability, hardware unreliability, software unreliability, Markov chains.

ACCESSION NO.	
NTIS	White Section <input checked="" type="checkbox"/>
DDI	Buff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION:	
BY:	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	AVAIL. and/or SPECIAL
A	

I. INTRODUCTION

Most of the past work on reliability modeling for digital systems has been centered around evaluation of the efficiency of redundancy techniques to achieve ultra-reliability. Triple Modular Redundancy [1 - 3], hybrid [4 - 7], stand-by [8 - 10], self-purging [11 - 13] and duplex [14, 15] redundancy techniques have been the focus of much attention. Recently, software tools have been developed [16, 17]. Most of these studies are directed towards reliability evaluation of computer systems intended for aeronautic and space applications where the cost of computer failure is so high as to allow the use of massive redundancy. However, there are many other applications, like banking and airline reservations systems, where massive redundancy techniques are economically unattractive but where continuous service is still a must. In the last few years there have been several attempts to provide general-purpose multiprocessor computers with high reliability and availability by capitalizing on the inherent redundancy of multiprocessors [18, 19]. They can provide uninterrupted service if each occurrence of a failure is detected and followed by a reconfiguration of the complete system such that some service is still available from the reconfigured system. This philosophy, to trade some computing power for continuous operation, is usually called graceful degradation [18].

Most of the reliability models for ultra-reliable systems are not well suited for the study of commercial systems that use graceful degradation. Ultra-reliable systems, like those used in airplanes or spaceships, are intended and designed to perform a mission of known length without any system failure. In-flight repair is usually impossible

and interruption of service for as short a time as a few minutes is usually unacceptable (e.g., if the interruption occurs during a computer controlled automatic landing). In contrast, commercial applications put more emphasis on availability. A system like an airline reservation system needs to be up most of the time. Interruptions of service are acceptable as long as they are quite short (a few minutes) and slower response can also be tolerated for brief periods. Commercial systems also differ substantially from ultra-reliable systems by their architecture. Commercial systems for safe computing are gracefully degradable multiprocessors with large software while ultra-reliable systems are in general specialized uniprocessors, protected from failures by external redundancy, and with very limited software.

This paper presents a general model for performance evaluation of gracefully degradable systems. The model is very general so that it can be applied to systems so different as Pluribus [20], C.mmp [21] or PRIME [22]. It differs substantially from the models developed by Borgerson [18] and Hayes [23]. Availability, down-time per system crash, percentage of time in degraded mode and overhead due to recovery and reconfiguration are analyzed. Both hardware and software unreliability are taken into account. Analysis of sensitivity to such parameters as the frequency of software tests leads to fine system tuning. Optimization completes the study.

II. DESCRIPTION OF GRACEFULLY DEGRADABLE SYSTEMS

II.1 GENERAL DESCRIPTION

Computing systems can be defined in broad terms as a set of hardware resources managed by software to provide users with some specific service.

The hardware is composed of several resources: processing, memories for storage, I/Os for communication with the outside world, and an intercommunication network (bus, crossbar switch, etc.) to provide interconnection between the elements. Each type of resource may contain several elements that provide the same kind of service; for example, there may be several CPUs and memory modules. The intercommunication network may also have some inherent redundancy (redundant busses, alternate disjoint paths, etc.).

Software is a major resource in any large computing system, both in terms of cost and as sources of failures [24]. Operating systems are responsible for management of the system so as to provide the best use and sharing of the resources among the users. In gracefully degradable systems, software is also responsible for fault diagnosis, recovery and hardware reconfiguration. Utility software, like compilers, file systems and text editors, is oriented more towards aiding the users than overall system management.

The services provided by computing systems are quite varied, from control of automatic processes up to the broad variety of services provided by large time sharing systems serving several hundred users. The

type and variety of services provided influence very significantly the overall system performance even with respect to reliability. Restricted applications can be satisfied with relatively simple software while some computation centers have operating systems in excess of one hundred thousand words and extremely complex internal organization (complicating recovery and reconfiguration).

II.2 FAILURES

Failures can be defined, in broad terms, as any event that breaks the regular system operation. Failures can be hardware malfunctions, software errors or occurrences of a situation that the software cannot handle (especially in complex real-time systems). In computer systems with large software, software unreliability is a major cause of failures [24] and, consequently, reliability models should take it into account.

One of the major characteristics of failures is their duration. Transient failures correspond to malfunctions that disappear after a short time (i.e., short, temporary hardware malfunctions during a burst of electromagnetic radiation or deadlock situations that are solved by killing one process). Permanent (also call hard or solid) failures correspond to permanent malfunctions that necessitate physical repair (or debugging).

The extent of a failure corresponds to the range of the corresponding malfunction. Failures can be catastrophic if the malfunction extends to the whole system (i.e., a power supply failure in a system with single supply affects the operation of all the hardware). In general, the extent of any failure is limited to one (or very few) element (i.e., one CPU for

most of the CPU-related failures, one memory module for most memory failures). It should be noted that the extent of a failure characterizes the physical range of the malfunction and not the range of the failure effects (e.g., a failure that occurs in a CPU may affect, if it goes undetected for some time, the validity of the data in memory).

II.3 DETECTION, RECOVERY AND RECONFIGURATION

The basic idea behind graceful degradation is to detect the failures as soon as they occur, determine their extent, logically disconnect the faulty element(s), reconfigure the remaining system as a useful one, try to recover from the failure effects on data integrity and resume normal operation in a degraded mode until the malfunction is repaired.

Detection is achieved by hardware (codes [25], memory protection, self-checking circuits [26], duplication of control mechanisms, time-out counters, etc.) by software (periodic test of the hardware, retry instructions, capability lists [27], performance monitors and schedulers for such problems as deadlocks of infinite loops, etc.) or eventually by the users and operators when all automatic detection methods have failed. Failures can be differentiated, with respect to detection, into two classes. The first class corresponds to the failures that are detected as soon as they occur (or, more precisely, by the first error they cause; that is to say, by the first deviation from correct operation). Failures of the first class are safe because they are detected before their effects propagate. The second class groups all other failures, which means all failures that are not detected upon the occurrence of their induced first

error. These failures are unsafe for they may cause data contamination or faulty controls before they are detected. Following such failures, it is necessary to assure the integrity of the data that have been contaminated during the detection latency [28] (the time period between the first error and the first detected error). This is of particular importance in multi-access data base systems.

After a detection occurs, it is necessary to determine the extent of the failure. Diagnostic programs and testing strategy can be used to localize the faulty elements.

According to the extent of failures and the range of their effects, it is possible to isolate the faulty element(s) and reconfigure the system in such a way that some useful service is still available. Reconfiguration involves such steps as possible relocation of the operating system in a fault-free memory, memory address remapping to isolate a faulty memory module or running an n -processor system as an $n-1$ processor system.

Recovery concerns the restoration of the overall system control, the restoration of the integrity of data (files, data bases, etc.) and the ordered restart of the system operation. The extent of the recovery process depends heavily on the type of applications. Systems oriented towards multi-access data base transactions need to guarantee high integrity of the data bases. Failures in telephone systems can result in the loss of some calls as long as the recovery is fast enough (i.e., to service those calls on the dialing). Restoration of data integrity (including operating system tables) can be achieved by roll-back and rerun, use of traces, updating from a redundant copy of the contaminated data or

program roll-aheads. Restoration of data integrity following unsafe failure is complicated by the fact that detection and diagnostic mechanisms may not differentiate between safe and unsafe failures and that the detection latency of unsafe failures is unknown.

II.4 PERFORMANCE

The major concern underlying the use of gracefully degradable systems is to provide continuous service even following occurrences of failures. Availability, defined as the probability the system is up, duration of the unavailability periods, average computing power (taking into account operation in degraded mode) and overhead due to recovery and reconfiguration are some of the major parameters that can be used to characterize gracefully degradable systems.

System failure can result from exhaustion of a resource (catastrophic failures or series of failures too close for repair to be completed) or from unsafe failures. During the detection latency of unsafe failures, the system is running but produces faulty results (which require job rerun after the failure has been detected). Duration of unavailability periods is heavily dependent upon the type of system crash (resource exhaustion or unsafe failures) and their associated parameters (repair time or detection latency plus reconfiguration and recovery overhead).

The average computing power is a function of the time spent in degraded mode, of repair times and overhead associated with recovery/reconfiguration.

III. GENERAL MODELING

III.1 SYSTEM PARTITIONING

Gracefully degradable systems can be partitioned into several independent resources. Each resource provides the system with a special type of service and is constituted by one or several elements that are functionally identical and that share the resource load. For example, the PRIME system (Figure 1) can be partitioned into seven resources: the resource for processing composed of five CPUs (CPU = processor plus dedicated map and I/O controller), a fast memory resource formed of 13 memory modules, a secondary memory resource (disc drives), an I/O device resource, two communication networks (the external access network and the CPU-memory network) and a software resource (operating system).

For the system to be operable, a certain degree of performance is required from each resource. This will be referred to as the minimum configuration (e.g., one processor, one memory and some I/Os). Overall system performance (computing power, availability, etc.) is directly dependent upon the performance of each resource. So, by analyzing each resource separately one can obtain a fairly accurate evaluation of the overall system performance.

III.2 MODEL FOR THE RESOURCES

Some resources, like the memory and processing resources, have highly parallel structures. They are formed by several identical, physically independent elements and the amount of service that can be obtained from such a resource is proportional to the number of fault-free elements. However,

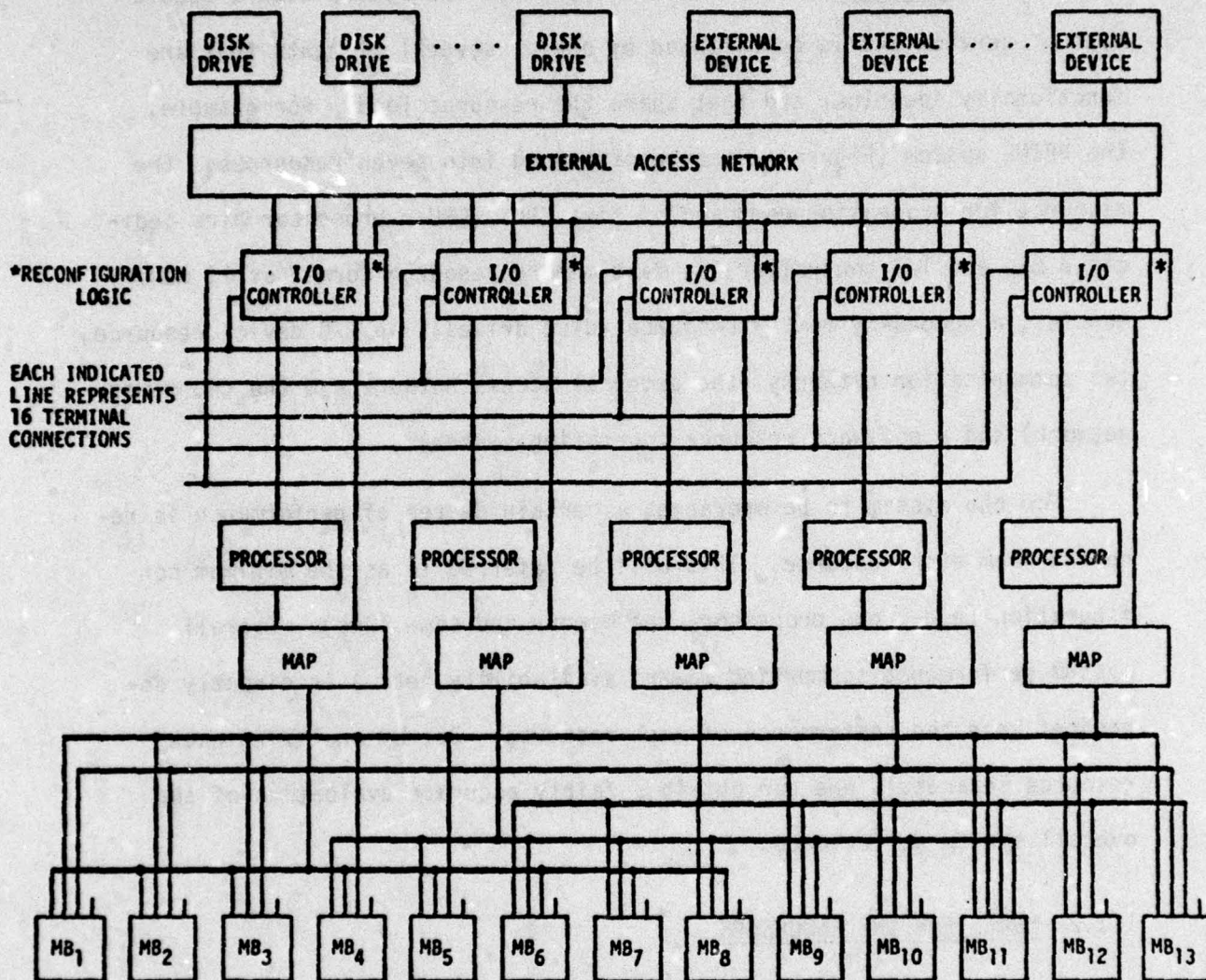


Fig. 1. A block diagram of the PRIME system.

resources like the interconnection networks (and software) lack such a well defined parallelism even though they may still have some inherent redundancy (not every failure is catastrophic). Network redundancy (number of paths between two nodes, minimal cuts, etc.) has been treated in detail [29, 30]. Redundancy in software is not very common. So, one may consider software as a resource with a single element for which identical copies are readily available from disc (the presence of copies does not correspond to redundancy for every copy presents the same defects). For the modeling of interconnection networks, one can find the average decrease in the information flow due to a failure. The relative value of the decrease gives an indication of the degree of parallelism and this can be used to model real networks as several independent fictitious subnetworks in parallel. For example, in the CPU-memory interconnection network in PRIME, a pessimistic analysis shows that, in average, a failure in the net disconnects one processor from the memories. So, one can approximately model such a network as five independent fictitious subnetworks in parallel. Each failure inside a subnetwork makes it totally inoperative and the transfer rate on the subnetwork is one fifth of that of the real network.

In the following, one will assume that a resource is formed by n identical and independent (failure-wise) elements. Out of these n elements, m need to be fault-free for the system to be able to run. Element failures are grouped into two classes according to detection. The failures that are detected on the first error they cause are called safe. The other failures are unsafe and characterized by the latency between the first error

and the first detected error. Following the detection of unsafe failures (most of the failures detected by periodic test), it is necessary to restore the integrity of the data that may have been contaminated. One will assume that the time needed to restore system integrity is directly proportional to the detection latency. The reasoning behind this assumption is that it will be necessary to check every action the system has been taking during the latency period. The proportionality coefficient, e (e for extent of recovery), is strongly dependent upon the type of applications the system is used for. Electronic reconfiguration of the hardware will be considered to be instantaneous.

The elements have the following characteristics:

λ = failure rate

μ = repair rate

α = probability that a failure be transient

c = probability that a failure be safe

$\frac{1}{v_d}$ = detection latency of the unsafe failures

e = multiplicative factor giving the recovery time from the latency

$v = \frac{v_d}{1+e}$ = rate of complete recovery following an unsafe failure (assumed to be a constant rate).

The state of a resource is fully defined by the number, x , of fault-free elements, and the number, y , of elements which either have undetected failures or are recovering from them. Such a state will be referred to a

$S_{x,y}$ and its steady state probability by $P_{x,y}$. The corresponding Markov chain and the transition probabilities are given in Figure 2.

The transition probabilities are quite explicit. Upon detection of a safe failure, the number of fault-free elements decreases by one if the failure is permanent and stays constant if the failure is transient. Upon occurrence of an unsafe failure, the number of fault-free elements is decreased by one and the number, y , of unsafe elements is increased by one. Unsafe failures are recovered from with rate v and the faulty module sent for repair. Unsafe transient failures may not be caught by the system detection mechanisms but only by the users (thus, a very large latency). To take into account the large overhead associated with verification of the data integrity after detection of such a failure, we assume that the overall effect will be similar to that of sending one element to repair.

The steady-state of this Markov chain can be obtained after some mathematical work:

$$P_{x,0} = \left[\frac{u(1+v)}{1 + u(1+v)} \right]^n \cdot \binom{n}{x} \cdot \left[\frac{1}{1+v} \right]^x \cdot \left[\frac{v}{1+v} \right]^{n-x}$$

$$u = \left[(1-\alpha) \cdot c + \frac{v}{\lambda} \right] \frac{1}{1-c} \quad v = \left[(1-\alpha) \cdot c + (1-c) \right] \frac{\lambda}{\mu} .$$

The resource (and thus the system) is unavailable when there are not enough fault-free elements to meet the system requirement or during the latency and recovery periods following unsafe failures. During latency periods, the resource produces faulty results and recovery periods

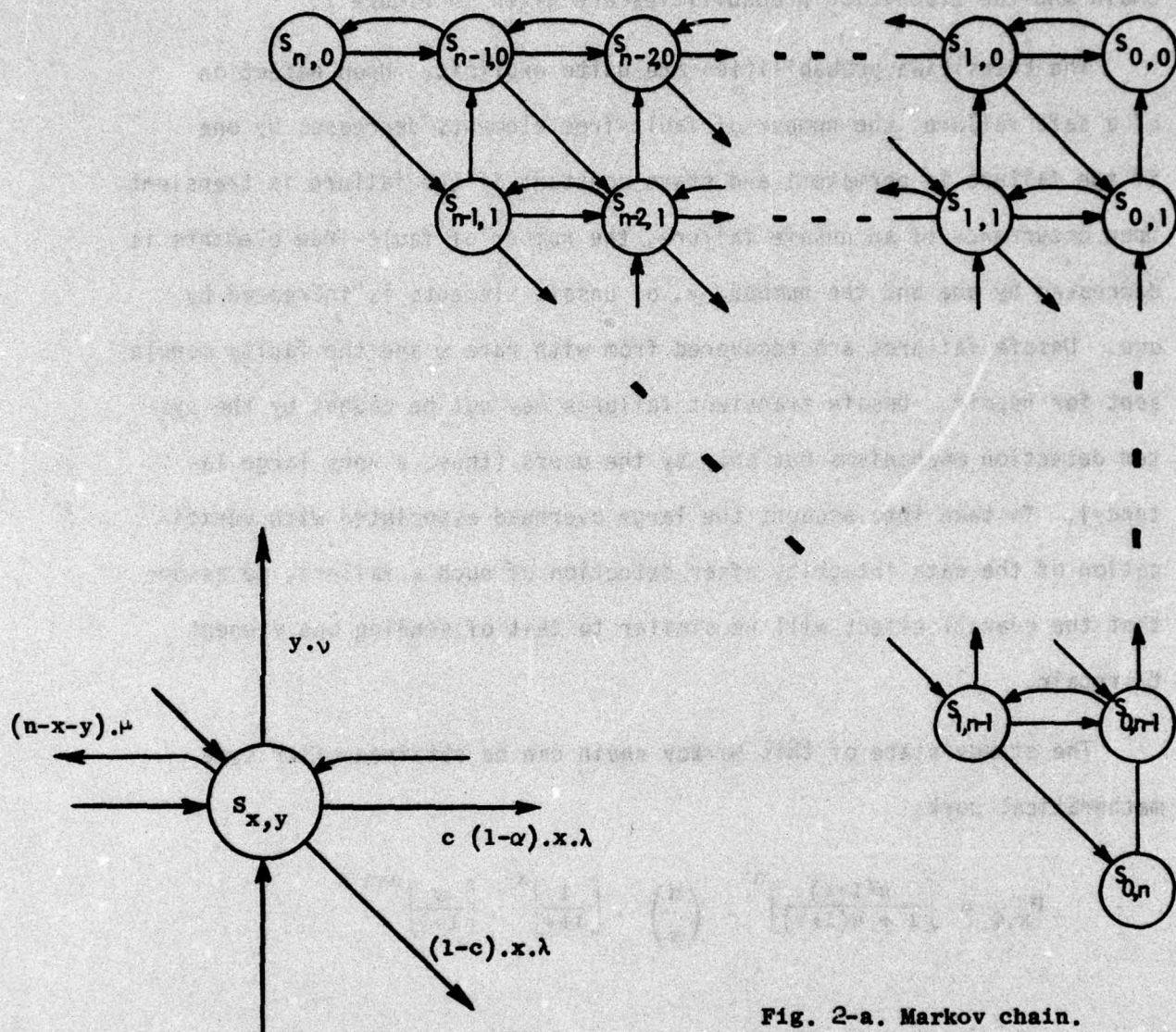


Fig. 2-a. Markov chain.

Fig. 2-b. Output transitions probabilities
from state $s_{x,y}$.

Fig. 2. Markov chain and transition probabilities.

are overhead (as far as users are concerned). So the resource availability, A , is

$$A = \sum_{x=n}^n P_{x,0}$$

$$A = \left[\frac{u(1+v)}{1+u(1+v)} \right]^n \cdot \sum_{x=n}^n \binom{n}{x} \cdot \left[\frac{1}{1+v} \right]^x \cdot \left[\frac{v}{1+v} \right]^{n-x}$$

$$A \approx 1 - n(1-c) \cdot \frac{\lambda}{v} - \binom{n}{m-1} \cdot \left[(1-c) \cdot \frac{\lambda}{\mu} \right]^{n-m+1}.$$

The unavailability due only to element exhaustion, U_e , is

$$U_e = \sum_{x=0}^{m-1} P_{x,0}$$

$$U_e = \left[\frac{u(1+v)}{1+u(1+v)} \right]^n \cdot \sum_{x=0}^{m-1} \binom{n}{x} \cdot \left[\frac{1}{1+v} \right]^x \cdot \left[\frac{v}{1+v} \right]^{n-x}.$$

The average number of unavailability periods by year (1 year = 1 time unit) is:

$$(1-c) \cdot \left[1 - (1-c) \cdot \frac{\lambda}{\mu} - n \cdot (1-c) \cdot \frac{\lambda}{v} \right] \cdot n \cdot \lambda$$

and the average duration of the unavailability periods is:

$$\frac{1}{v} \cdot \left[1 + (1-c.\alpha) \cdot \frac{\lambda}{\mu} + n \cdot (1-c) \cdot \frac{\lambda}{v} \right] .$$

III.3 OPTIMIZATION OF RESOURCES

As it can be seen from the equations, availability and throughput are dependent upon the number, n , of elements. When the resource can provide some service with only one fault-free element ($m=1$), the value for n , n^* , which maximizes the availability, can be simply obtained as:

$$n^* = \frac{-\text{Log}\left[1 - \frac{1}{1-c} \cdot \frac{v}{\lambda} \cdot \text{Log}\left((1-c.\alpha) \cdot \frac{\lambda}{\mu}\right)\right]}{\text{Log}\left[(1-c.\alpha) \cdot \frac{\lambda}{\mu}\right]} .$$

When m is larger than 1, the analytical solution for n^* is quite complex. However, if one expresses n^* as a function of m , $n^* = (1+\epsilon) \cdot m$, one can get an approximation by using the approximation between the binomial and normal distribution.

$$\epsilon = \frac{n^* - m}{m} \approx (1-c.\alpha) \cdot \frac{\lambda}{\mu} + \sqrt{2 \cdot (1-c.\alpha) \cdot \frac{\lambda}{\mu} \cdot \text{Log}\left[\frac{1}{1-c} \cdot \frac{v}{\lambda} \cdot \sqrt{\frac{1}{1-c.\alpha} \cdot \frac{1}{2 \cdot \pi \cdot m} \cdot \frac{\mu}{\lambda}}\right]} .$$

Table 1 lists a few values of n^* as a function of the element characteristics and the value of m . It is striking to note that availability is maximum when the number of elements, n , is quite close to the minimum, m . It is also worthy to note that the optimum is largely insensitive to the characteristics of the unsafe failures. The similarity between these results and those concerning optimization of hybrid [9] and stand-by [31] systems is quite interesting.

The response throughput, E , can also be maximized. When m is equal to 1, the optimal n^{**} is

$$n^{**} = \left[(1-x) \cdot c + \frac{v}{\lambda} \right] \frac{1}{1-c}$$

which, as expected, is far larger than n^* . However, the maximization of the average throughput of an element, $\frac{E}{n}$, gives a result very close to n^* . As the number of elements is increased beyond n^* , both the availability and the average throughput of each element decreases.

$1/v \backslash m$	4	100
10^{-3} s.	4.45	111.
1 s.	4.40	109.6
60 s.	4.36	108.7
3600 s.	4.32	107.7

Table 1-a. $\begin{cases} \alpha = .8 \\ c = .9 \end{cases}$

$1/v \backslash m$	4	100
10^{-3} s.	4.6	113.7
1 s.	4.8	120.
60 s.	4.9	123.1
3600 s.	4.6	115.1

Table 1-b. $\begin{cases} \alpha = 0 \\ c = .9 \end{cases}$

$1/v \backslash m$	4	100
10^{-3} s.	4.5	111.5
1 s.	4.4	110.6
60 s.	4.4	109.2
3600 s.	4.4	108.3

Table 1-c. $\begin{cases} \alpha = .8 \\ c = .99 \end{cases}$

$1/v \backslash m$	4	100
10^{-3} s.	4.6	115.3
1 s.	4.9	121.1
60 s.	5.0	124.1
3600 s.	4.7	116.1

Table 1-d. $\begin{cases} \alpha = 0 \\ c = .99 \end{cases}$

Table 1. Optimal values of n, n^* , under the following conditions: $\lambda = 10^{-4}/h$, $\mu = .1/h$.

IV. EXAMPLE

IV.1 DESCRIPTION

To illustrate the model, we will apply it to an example. The system modeled will be a simplified version of the PRIME system [18, 22] (Fig. 1): five processors, 13 memory modules, 15 discs, an external access network modeled as ten independent switches, and some external devices (we will assume six identical units). We will neglect the unreliability of the processor-memory interconnection network. Because of the unavailability of failure-related data concerning the software run on the PRIME system, we will assume for this example, that the system is running a widely used operating system (e.g., a combination of OS/MVT, HASP plus some editors, for which some failure-related data are available).

We will further assume that the hardware is equipped with some error-detection mechanism (parity checking). Almost all the failures detected by parity are detected on the first few errors, so one can assume that all failures detected by hardware are safe. The system is also periodically tested by software. Failures detected by software are likely to be unsafe (they have bypassed the hardware detection mechanisms and are detected only by extensive periodic tests). The probability, c , that failure be detected by the hardware fault detection mechanism will be taken as .90 (the use of codes should guarantee such a coverage).

The frequency of the software periodic tests will be 1 per minute. Using the data relevant to test efficiency in [18], the conditional probability that such a test detects a failure given a failure is .90. This leads (assuming independence of successive tests) to an average detection

latency of .6 minutes. Assuming conservatively that recovery takes twice as long, the average unavailability is around 2 minutes long.

Data gathered during an eight month period on an IBM 360/91 running OS/MVT, HASP and text editor "WYLBUR" indicate that the meantime between software failures is fairly constant and runs around 70 hours. Two thirds of the failures cause system crash and necessitate restart (approximately 15 minutes). The other third causes system degradation for around 15 minutes. So, for the modeling of such a software, one can make the assumption that there are two software resources. The O.S. forms a resource by itself (one element, $\lambda = 10^{-2}/h$, $\mu = 4/h$). The rest of the software will be divided into two independent elements ($\lambda = 2.5 \cdot 10^{-3}/h$, $\mu = 4/h$, $c = 1$). A summary of the system parameters is given in Table 2.

IV.2 RESULTS

The results of the study of this system can be summerized as follows:

- System availability = 99.74%.
- Software failures contribute 97% of all unavailability.
- Meantime between system crashes = 57.5 hours.
- Meantime between crashes due to hardware = 418 hours.
- Average down time per crash = 14 minutes.
- Average down time per hardware-caused crash = 2 minutes.
- 17.4% of the time, the system operates in degraded mode.

Table 2. Parameters for the system considered in the example.

Resources Parameters	Proc.	Mem.	Comm. net	Discs	I/O's	Software	
						O.S.	other
n	5	13	10	15	6	1	2
m	3	8	6	10	3	1	1
$\lambda/10^5 \text{ h.}$	15	4	1	30	300	10^3	250
μ/h	.1	.5	.5	.2	.2	4	4
α	.1	.1	.1	.1	.1	0	0
c	.9	.9	.9	.9	.9	1	1
v/h	30	30	30	30	30	-	-

- Less than 1% of all hardware unavailability is due to resource exhaustion.
- The average processing power is 4.97 processors,
12.98 memories,
10.00 switches,
14.69 discs, and
5.51 output devices.

When the unreliability of the input/output devices is not taken into account, the meantime between crashes due to hardware is around 1700 hours and the availability is .99998. These results agree reasonably well with those obtained by Borgerson [18]. So, the unreliability of the input/output device is the major factor limiting the hardware availability. It should also be noted that not every failure in the input/output devices is safe. For example, in the 360/67 system at Stanford, out of 61 line printer failures, five caused system crashes.

It may be interesting to look at the effects of the hardware detection mechanisms and the frequency of periodic tests on the availability of the hardware (Figures 3 and 4).

The introduction of hardware detection mechanisms is extremely beneficial, even if they are fairly inefficient (small values of c). But, even with the best mechanisms, availability is limited. The frequency with which software tests are run has the same effect (Figure 4). Frequent testing is beneficial but, past a certain limit, the incremental availability gain may not compensate for the increase in overhead.

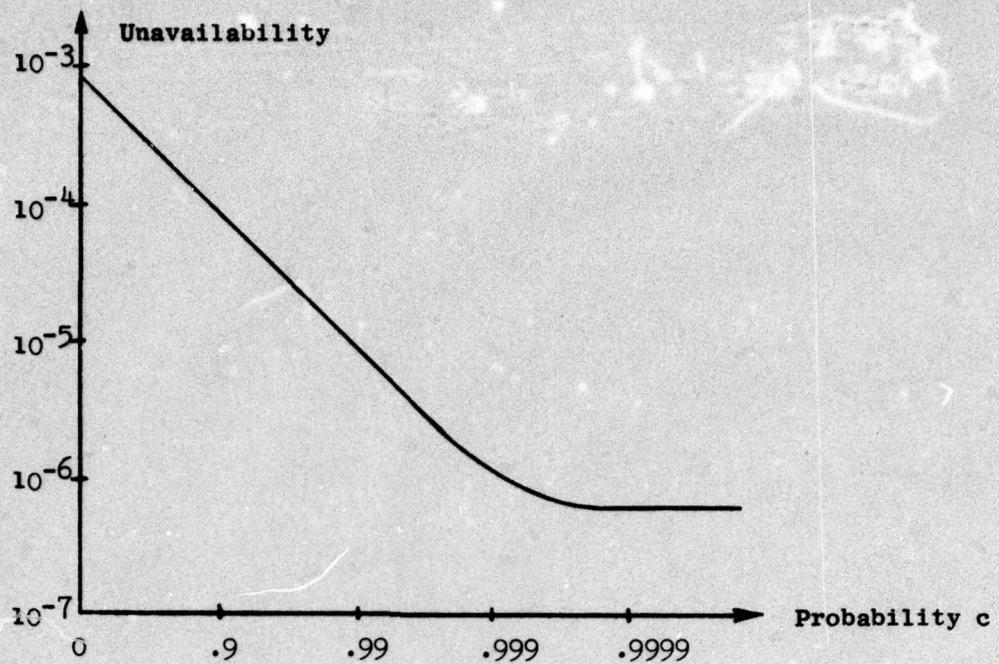


Fig. 3. Variation of the unavailability as function of the probability c .

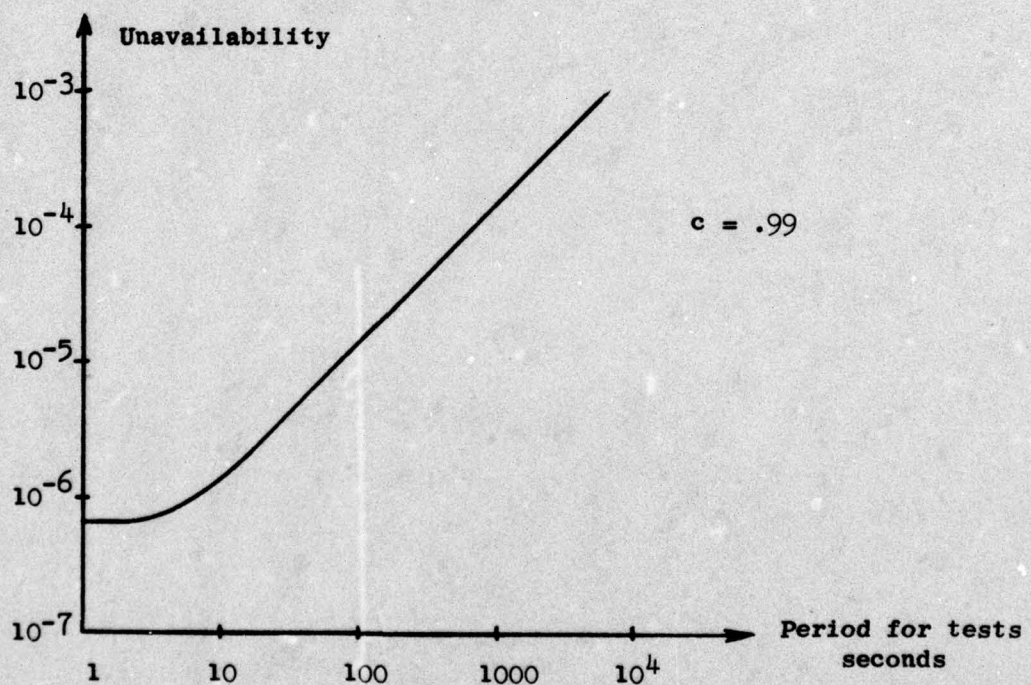


Fig. 4. Variation of the unreliability as function of the test period.

V. CONCLUSIONS

The previous example clearly shows the benefits of graceful degradation at the hardware level. Lengthy downtimes due to repairs are eliminated. Systems operate longer without interruption, unavailability periods are quite short and overall processing power is not affected much by failures. This type of smooth operation over indefinite periods of time may be extremely well suited for a large range of applications. However, gracefully degrading systems are as varied as their applications so that an universal tool for performance evaluation is highly desirable.

The model presented here is based on a general description of the principles behind graceful degradation. It is oriented towards performance evaluation and not only reliability analysis. Classification of failures between safe and unsafe allows one to model the different methods of fault detection. Detection latencies are useful to assess the damages that a failure can create before it is detected. The importance accorded to data integrity is taken into account in modeling the duration of the recovery processes. The partitioning of systems into resources simplify considerably the analysis and provides a way to take software unreliability into account.

Study of the model shows that much attention should be given to optimization. Indeed, by increasing the degree of redundancy beyond a certain limit some of the performance decreases; for example, availability and the ratio of efficiency by cost. The model also points to the

practical interest of hardware fault-detection mechanisms and frequent software tests and to some of the trade-offs.

One of the most interesting problems associated with gracefully degrading systems is the structure of the software when it is considered as a resource which can, and indeed does, suffer failures. It is hoped that some investigation of this problem may be carried out in the near future.

VI. REFERENCES

1. [Von Neuman, 1965] Von Neuman, J., "Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components," Automata Studies (Annals of Mathematical Studies), pp. 43-98, C.E. Shannon and J. McCarthy, Editors, Princetown University Press, Princeton, New Jersey, 1965.
2. [Lyons & Vanderkulk, 1962] Lyons, R.E. and W. Vanderkulk, "The Use of Triple Modular Redundancy to Improve Computer Reliability," IBM Jour. of Res. Development, Vol. 6, pp. 200-209, 1962.
3. [Abraham & Siewiorek, 1974] Abraham, J.A. and D.P. Siewiorek, "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," IEEE Trans. on Computers, Vol. C-23, pp. 682-692, July 1974.
4. [Mathur & Avizienis, 1970] Mathur, F.P. and A. Avizienis, "Reliability Analysis and Architecture of a Highly Redundant Digital System: Generalized Triple Modular Redundancy with Repair," Proc. SJCC, Vol. 26, pp. 375-383, 1970.
5. [Roth, et al., 1967] Roth, J.P., W.G. Bouricius, W.C. Carter and P.R. Schneider, "Phase II of an Architectural Study for a Self-Repairing Computer," SAMSO TR 67-106, November 1967.
6. [Mathur & de Souza, 1975] Mathur, F.P. and P.T. de Souza, "Reliability Modeling and Analysis of General Modular Redundant Systems," IEEE Trans. Reliability, Vol. R-24, pp. 296-299, December 1975.
7. [Knox-Seith, 1963] Knox-Seith, J.K., "A Redundancy Technique for Improving the Reliability of Digital Systems," Tech. Rpt. No. 4816-1, Stanford Electronics Labs, Stanford University, Stanford, California, December 1963.
8. [Avizienis, 1967] Avizienis, A., "Design of Fault-Tolerant Computers," FJCC, Vol. 31, pp. 733-743, 1967.
9. [Bouricius, et al., 1969] Bouricius, W.G., W.C. Carter and P.R. Schneider, "Reliability Modeling Techniques for Self-Repairing Computer Systems," Proc. ACM 1969 Annual Conf., pp. 295-305; also, IBM Rpt. No. R.C-2378.

10. [Goldberg, et al., 1966] Goldberg, J., K.N. Levitt and R.A. Short, "Techniques for Realization of Ultra-Reliable Spaceborn Computers," Final Rpt., Phase I, SRI Project 5580, Stanford Research Institute, Menlo Park, California, September 1966.
11. [Pierce, 1962] Pierce, W.H., "Adaptive Vote-Takers Improve the Use of Redundancy," Redundancy Techniques for Computing Systems, pp. 229-250, Spartan Books, Washington, D.C., 1962.
12. [Chandy & Ramamoorthy, 1972] Chandy, K.N. and C.V. Ramamoorthy, "A Framework for Hardware-Software Tradeoffs in the Design of Fault-Tolerant Computers," FJCC, AFIPS, pp. 55-63, 1972.
13. [Losq, 1976] Losq, J., "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," IEEE Trans. on Computers, Vol. C-25, pp. 269-278, June 1976.
14. [Arnold, 1973] Arnold, T.F., "The Concept of Coverage and Its Effect on the Reliability Model of a Repairable System," IEEE Trans. on Computers, Vol. C-22, pp. 251-254, March 1973.
15. [Gaver, 1963] Gaver, D.P., "Time to Failure and Availability of Paralleled Systems with Repair," IEEE Trans. on Reliability, Vol. R-12, pp. 30-38, June 1963.
16. [Masreliez & Bjurman, 1976] Masreliez, C.J. and B.E. Bjurman, "Fault-Tolerant System Reliability Modeling/Analysis," Boeing Corp., Seattle, Washington, June 1976.
17. [Beaufils, et al., 1974] Beaufils, R., J.L. Paul and R. Troy, "Système d'Evaluation Globale de Multiprocesseurs Autoréparables," Rapport 1 and 2, Contrat DRME 73/070, LAAS, University of Toulouse, Toulouse, France, 1974.
18. [Borgerson & Freitas, 1975] Borgerson, B.R. and R.F. Freitas, "A Reliability Model for Gracefully Degrading and Standby Sparring Systems," IEEE Trans. on Computers, Vol. C-24, pp. 517-525, May 1975.
19. [Wensley, 1972] Wensley, J.M., "SIFT - Software Implemented Fault Tolerance," FJCC Proc., pp. 243-253, 1972.
20. [Ornstein, et al., 1975] Ornstein, S.M., W.R. Crowther, M.F. Krale, A. Michel and F.E. Heart, "Pluribus - A Reliable Multiprocessor" Proc. AFIPS, pp. 551-559, 1975.

21. [Wulf & Bell, 1972] Wulf, W.A. and C.G. Bell, "C.mmp--A Multi-Mini Computer," Proc. AFIPS, 1972 FJCC, pp. 756-777, 1972.
22. [Baskin, et al., 1972] Baskin, M.B., B.R. Borgerson and R. Roberts, "PRIME - A Modular Architecture for Terminal Oriented Systems," Proc. SJCC, Vol. 40, pp. 431-437, AFIPS Press, Montvale, New Jersey, 1972.
23. [Hayes, 1976] Hayes, J.P., "A Graph Model for Fault Tolerant Computing," IEEE Trans. on Computers, Vol. C-25, pp. 875-884, September 1976.
24. [Brooks, 1975] Brooks, F.P.Jr., The Mythical Man-Month, Editor, Addison-Wesley, Reading, Massachusetts, 1975.
25. [Peterson & Weldon, 1972] Peterson, W.W. and E.J. Weldon, Jr., Error-Correcting Codes, M.I.T. Press, Cambridge, Massachusetts, 1972.
26. [Wakerly, 1974] Wakerly, J.F., "Partially Self-Checking Circuits and Their Use in Performing Logic Operations," IEEE Trans. on Computers, Vol. C-23, pp. 650-666, July 1974.
27. [Lampson, 1971] Lampson, B.W., "Protection," Proc. Fifth Annual Princeton Conf. on Information Sciences and Systems, pp. 437-443, Dept. of Elect. Eng., Princeton University, Princeton, New Jersey, March 1971.
28. [Shedletsky & McCluskey, 1976] Shedletsky, J.J. and E.J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit," IEEE Trans. on Computers, Vol. C-25, pp. 655-659, June 1976.
29. [Murty, 1968] Murty, U.S.R., "On Some Extremal Graphs," Acta Mathematica Academiae Scientiarum Hungaricae, Tomus 19, pp. 69-74, 1968.
30. [Berge, 1970] Berge, C., Graphes et Hypergraphes, Dunot, Editor, Paris, France, 1970.
31. [Losq, 1975] Losq, J., "Influence of Fault Detection and Switching Mechanisms on the Reliability of Stand-by Systems," Proc. FTC-5, pp. 81-86, Paris, France, 1975.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER Technical Note No. 103		2. GOVT ACCESSION NO.	
4. TITLE (and Subtitle) Effects of Failures on Performance of Gracefully Degradable Systems		5. TYPE OF REPORT & PERIOD COVERED Technical Note	
7. AUTHOR(s) Jacques Losq		6. PERFORMING ORG. REPORT NUMBER 15	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Digital Systems Laboratory Stanford University Stanford, CA 94305		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0601, NSF MCS-76-05327	
11. CONTROLLING OFFICE NAME AND ADDRESS Sponsored Projects Office Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Su 7151 and 7106	
14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office) Stanford Electronics Laboratories Stanford University Stanford, CA 94305		12. REPORT DATE Dec 1976	
16. DISTRIBUTION STATEMENT (of this report) This document has been approved for public release and sale; its distribution is unlimited.		13. NO. OF PAGES 28	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from report)		15. SECURITY CLASS. (of this report) Unclassified	
18. SUPPLEMENTARY NOTES		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE 12 31p	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) availability hardware unreliability digital systems Markov chains fault-tolerance software unreliability graceful degradation			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The recent development of multiprocessor systems that offer resistance to faults by gracefully degrading after a failure opens vast new ranges of applications for fault tolerance and high reliability. The paper presents a general model for the evaluation of such systems. It takes into account the internal structure of the hardware, the characteristics of the various detection mechanisms, the unreliability of the software and even the type of applications these systems for which these systems are used. It provides many measures of the systems' performance such as: availability, meantime between crashes, average processing power and proportion of time spent in degraded mode. System optimization gives the best values for the			

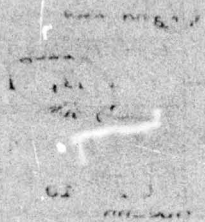
DD FORM 1473
1 JAN 73
EDITION OF 1 NOV 65 IS OBSOLETE

408071

2 pages
Page
JAC

20. Abstract (continued)

number of processors, memories,..., and shows the trade-offs between hardware and software fault-detection mechanisms. The model is illustrated by a concrete example.



JSEP REPORTS DISTRIBUTION LIST

Department of Defense

Director
National Security Agency
Attn: Dr. T. J. Beahn
Fort George G. Meade, MD 20755

Defense Documentation Center (12)
Attn: DDC-TCA (Mrs. V. Caponio)
Cameron Station
Alexandria, VA 22314

Assistant Director
Electronics and Computer Sciences
Office of Director of Defense
Research and Engineering
The Pentagon
Washington, D.C. 20315

Defense Advanced Research
Projects Agency
Attn: (Dr. R. Reynolds)
1400 Wilson Boulevard
Arlington, VA 22209

Department of the Army

Commandant
US Army Air Defense School
Attn: ATSAD-T-CSM
Fort Bliss, TX 79916

Commander
US Army Armament R&D Command
Attn: DRSAR-RD
Dover, NJ 07801

Commander
US Army Ballistics Research Lab.
Attn: DRXRD-BAD
Aberdeen Proving Ground
Aberdeen, MD 21005

Commandant
US Army Command and
General Staff College
Attn: Acquisitions, Library Div.
Fort Leavenworth, KS 66027

Commander
US Army Communication Command
Attn: CC-OPS-PD
Fort Huachuca, AZ 85613

Commander
US Army Materials and
Mechanics Research Center
Attn: Chief, Materials Sci. Div.
Watertown, MA 02172

Commander
US Army Materiel Development
and Readiness Command
Attn: Technical Lib., Rm. 7S 35
5001 Eisenhower Avenue
Alexandria, VA 22333

Commander
US Army Missile R&D Command
Attn: Chief, Document Section
Redstone Arsenal, AL 35809

Commander
US Army Satellite Communications
Agency
Fort Monmouth, NJ 07703

Director
US Army Signals Warfare Laboratory
Attn: DELSW-OS
Arlington Hall Station
Arlington, VA 22212

Project Manager
ARTADS
EAI Building
West Long Branch, NJ 07764

NOTE: One (1) copy to each addressee unless otherwise indicated.

Commander/Director
Atmospheric Sciences Lab. (ECOM)
Attn: DRSEL-BL-DD
White Sands Missile Range, NM 88002

Commander
US Army Electronics Command
Attn: DRSEL-NL-O
(Dr. H. S. Bennett)
Fort Monmouth, NJ 07703

Director
TRI-TAC
Attn: TT-AD (Mrs. Briller)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-CT-L (Dr. R. E. Buser)
Fort Monmouth, NJ 07703

Director
Electronic Warfare Lab. (ECOM)
Attn: DRSEL-WL-MY
White Sands Missile Range, NM 88002

Executive Secretary, TAC/JSEP
US Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709

Commander
Frankford Arsenal
Deputy Director
Pitman-Dunn Laboratory
Philadelphia, PA 19137

Project Manager
Ballistic Missile Defense
Program Office
Attn: DACS-DMP (Mr. A. Gold)
1300 Wilson Boulevard
Arlington, VA 22209

Commander
Harry Diamond Laboratories
Attn: Mr. John E. Rosenberg
2800 Powder Mill Road
Adelphi, MD 20783

HQDA (DAMA-ARZ-A)
Washington, D.C. 20310

Commander
US Army Electronics Command
Attn: DRSEL-TL-E (Dr. J. A. Kohn)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-TL-EN
(Dr. S. Kroenenberg)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-NL-T (Mr. R. Kulinyi)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-NL-B (Dr. E. Lieblein)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-TL-MM (Mr. N. Lipetz)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-RD-O (Dr. W. S. McAfee)
Fort Monmouth, NJ 07703

Director
Night Vision Laboratory
Attn: DRSEL-NV-D
Fort Belvoir, VA 22060

Col. Robert Noce
Senior Standardization Representative
US Army Standardization Group, Canada
Canadian Force Headquarters
Ottawa, Ontario, Canada KIA)K2

Commander
US Army Electronics Command
Attn: DRSEL-NL-B (Dr. D. C. Pearce)
Fort Monmouth, NJ 07703

Commander
Picatinny Arsenal
Attn: SMUPA-TS-T-S
Dover, NJ 07801

Commander
US Army Electronics Command
Attn: DRSEL-NL-RH-1
(Dr. F. Schwering)
Fort Monmouth, NJ 07703

Commander
US Army Electronics Command
Attn: DRSEL-TL-I
(Dr. C. G. Thornton)
Fort Monmouth, NJ 07703

US Army Research Office (3)
Attn: Library
P. O. Box 12211
Research Triangle Park, NC 27709

Director
Division of Neuropsychiatry
Walter Reed Army Institute
of Research
Washington, D.C. 20012

Commander
White Sands Missile Range
Attn: STEWS-ID-R
White Sands Missile Range, NM 88002

Department of the Air Force

Mr. Robert Barrett
RADC/ETS
Hanscom AFB, MA 01731

Dr. Carl E. Baum
AFWL (ES)
Kirtland AFB, NM 87117

Dr. E. Champagne
AFAL/DH
Wright-Patterson AFB, OH 45433

Dr. R. P. Dolan
RADC/ETSD
Hanscom AFB, MA 01731

Mr. W. Edwards
AFAL/TE
Wright-Patterson AFB, OH 45433

Professor R. E. Fontana
Head, Dept. of Electrical Engineering
AFIT/ENE
Wright-Patterson AFB, OH 45433

Dr. Alan Garscadden
AFAPL/POD
Wright-Patterson AFB, OH 45433

USAF European Office of
Aerospace Research
Attn: Major J. Gorrell
Box 14, FPO, New York 09510

LTC Richard J. Gowen
Department of Electrical Engineering
USAF Academy, CO 80840

Mr. Murray Kesselman (ISCA)
Rome Air Development Center
Griffiss AFB, NY 13441

Dr. G. Knausenberger
Air Force Member, TAC
Air Force Office of Scientific
Research, (AFSC) AFSOR/NE
Bolling Air Force Base, DC 20332

Dr. L. Kravitz
Air Force Member, TAC
Air Force Office of Scientific
Research, (AFSC) AFSOR/NE
Bolling Air Force Base, DC 20332

Mr. R. D. Larson
AFAL/DHR
Wright-Patterson AFB, OH 45433

Dr. Richard B. Mack
RADC/ETER
Hanscom AFB, MA 01731

Mr. John Mottsmith (MCIT)
HQ ESD (AFSC)
Hanscom AFB, MA 01731

Dr. Richard Picard
RADC/ETSL
Hanscom AFB, MA 01731

Dr. J. Ryles
Chief Scientist
AFAL/CA
Wright-Patterson AFB, OH 45433

Dr. Allan Schell
RADC/ETE
Hanscom AFB, MA 01731

Mr. H. E. Webb, Jr. (ISCP)
Rome Air Development Center
Griffiss AFB, NY 13441

LTC G. Wepfer
Air Force Office of Scientific
Research, (AFSC) AFOSR/NP
Bolling Air Force Base, DC 20332

LTC G. McKemie
Air Force Office of Scientific
Research, (AFSC) AFOSR/NM
Bolling Air Force Base, DC 20332

Department of the Navy

Dr. R. S. Allgaier
Naval Surface Weapons Center
Code WR-303
White Oak
Silver Spring, MD 20910

Naval Weapons Center
Attn: Code 5515, H. F. Blazek
China Lake, CA 93555

Dr. H. L. Blood
Technical Director
Naval Undersea Center
San Diego, CA 95152

Naval Research Laboratory
Attn: Code 5200, A. Brodzinsky
4555 Overlook Avenue, SW
Washington, D.C. 20375

Naval Research Laboratory
Attn: Code 7701, J. D. Brown
4555 Overlook Avenue, SW
Washington, D.C. 20375

Naval Research Laboratory
Attn: Code 5210, J. E. Davey
4555 Overlook Avenue, SW
Washington, D.C. 20375

Naval Research Laboratory
Attn: Code 5460/5410, J. R. Davis
4555 Overlook Avenue, SW
Washington, D.C. 20375

Naval Ocean Systems Center
Attn: Code 75, W. J. Dejka
271 Catalina Boulevard
San Diego, CA 92152

Naval Weapons Center
Attn: Code 601, F. C. Essig
China Lake, CA 93555

Naval Research Laboratory
Attn: Code 5510, W. L. Faust
4555 Overlook Avenue, SW
Washington, D.C. 20375

Naval Research Laboratory
Attn: Code 2627, Mrs. D. Folen
4555 Overlook Avenue, SW
Washington, D.C. 20375

Dr. Robert R. Fossum
Dean of Research
Naval Postgraduate School
Monterey, CA 93940

Dr. G. G. Gould
Technical Director
Naval Coastal System Laboratory
Panama City, FL 32401

Naval Ocean Systems Center
Attn: Code 7203, V. E. Hildebrand
271 Catalina Boulevard
San Diego, CA 92152

Naval Ocean Systems Center
Attn: Code 753, P. H. Johnson
271 Catalina Boulevard
San Diego, CA 92152

Donald E. Kirk
Professor and Chairman
Electronic Engineer, SP-304
Naval Postgraduate School
Monterey, CA 93940

Naval Air Development Center
Attn: Code 01, Dr. R. K. Lobb
Johnsville
Warminster, PA 18974

Naval Research Laboratory
Attn: Code 5270, B. D. McCombe
4555 Overlook Avenue, SW
Washington, D.C. 20375

Capt. R. B. Meeks
Naval Sea Systems Command
NC #3
2531 Jefferson Davis Highway
Arlington, VA 20362

Dr. H. J. Mueller
Naval Air Systems Command
Code 310
JP #1
1411 Jefferson Davis Highway
Arlington, VA 20360

Dr. J. H. Mills, Jr.
Naval Surface Weapons Center
Electronics Systems Department
Code DF
Dahlgren, VA 22448

Naval Ocean Systems Center
Attn: Code 702, H. T. Mortimer
271 Catalina Boulevard
San Diego, CA 92152

Naval Air Development Center
Attn: Technical Library
Johnsville
Warminster, PA 18974

Naval Ocean Systems Center
Attn: Technical Library
271 Catalina Boulevard
San Diego, CA 92152

Naval Research Laboratory
Underwater Sound Reference Division
Technical Library
P. O. Box 8337
Orlando, FL 32806

Naval Surface Weapons Center
Attn: Technical Library
Code DX-21
Dahlgren, VA 22448

Naval Surface Weapons Center
Attn: Technical Library
Building 1-330, Code WX-40
White Oak
Silver Spring, MD 20910

Naval Training Equipment Center
Attn: Technical Library
Orlando, FL 32813

Naval Undersea Center
Attn: Technical Library
San Diego, CA 92152

Naval Underwater Systems Center
Attn: Technical Library
Newport, RI 02840

Office of Naval Research
Electronic and Solid State
Sciences Program (Code 427)
800 North Quincy Street
Arlington, VA 22217

Office of Naval Research
Mathematics Program (Code 432)
800 North Quincy Street
Arlington, VA 22217

Office of Naval Research
Naval Systems Division
Code 220/221
800 North Quincy Street
Arlington, VA 22217

Director
Office of Naval Research
New York Area Office
715 Broadway, 5th Floor
New York, NY 10003

Office of Naval Research
San Francisco Area Office
One Hallidie Plaza, Suite 601
San Francisco, CA 94102

Director
Office of Naval Research
Branch Office
495 Summer Street
Boston, MA 02210

Director
Office of Naval Research
Branch Office
536 South Clark Street
Chicago, IL 60605

Director
Office of Naval Research
Branch Office
1030 East Green Street
Pasadena, CA 91101

Mr. H. R. Riedl
Naval Surface Weapons Center
Code WR-34
White Oak Laboratory
Silver Spring, MD 20910

Naval Air Development Center
Attn: Code 202, T. J. Shopple
Johnsville
Warminster, PA 18974

Naval Research Laboratory
Attn: Code 5403, J. E. Shore
4555 Overlook Avenue, SW
Washington, D.C. 20375

A. L. Slafkovsky
Scientific Advisor
Headquarters Marine Corps
MC-RD-1
Arlington Annex
Washington, D.C. 20380

Harris B. Stone
Office of Research, Development,
Test and Evaluation
NOP-987
The Pentagon, Room 5D760
Washington, D.C. 20350

Mr. L. Sumney
Naval Electronics Systems Command
Code 3042, NC #1
2511 Jefferson Davis Highway
Arlington, VA 20360

David W. Taylor
Naval Ship Research and
Development Center
Code 522.1
Bethesda, MD 20084

Naval Research Laboratory
Attn: Code 4105, Dr. S. Teitler
4555 Overlook Avenue, SW
Washington, D.C. 20375

Lt. Cdr. John Turner
NAVMAT 0343
CP #5, Room 1044
2211 Jefferson Davis Highway
Arlington, VA 20360

Naval Ocean Systems Center
Attn: Code 746, H. H. Wieder
271 Catalina Boulevard
San Diego, CA 92152

Dr. W. A. Von Winkle
Associate Technical Director
for Technology
Naval Underwater Systems Center
New London, CT 06320

Dr. Gernot M. R. Winkler
Director, Time Service
US Naval Observatory
Massachusetts Avenue at
34th Street, NW
Washington, D.C. 20390

Other Government Agencies

Dr. Howard W. Etzel
Deputy Director
Division of Materials Research
National Science Foundation
1800 G Street
Washington, D.C. 20550

Mr. J. C. French
National Bureau of Standards
Electronics Technology Division
Washington, D.C. 20234

Dr. Jay Harris
Program Director
Devices and Waves Program
National Science Foundation
1800 G Street
Washington, D.C. 20550

Los Alamos Scientific Laboratory
Attn: Reports Library
P. O. Box 1663
Los Alamos, NM 87544

Dr. Dean Mitchell
Program Director
Solid-State Physics
Division of Materials Research
National Science Foundation
1800 G Street
Washington, D.C. 20550

Mr. F. C. Schwenk, RD-T
National Aeronautics and
Space Administration
Washington, D.C. 20546

M. Zane Thornton
Deputy Director, Institute for
Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

Nongovernment Agencies

Director
Columbia Radiation Laboratory
Columbia University
538 West 120th Street
New York, NY 10027

Director
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

Director of Laboratories
Division of Engineering and
Applied Physics
Harvard University
Pierce Hall
Cambridge, MA 02138

Director
Electronics Research Center
The University of Texas
Engineering-Science Bldg. 112
Austin, TX 78712

Director
Electronics Research Laboratory
University of California
Berkeley, CA 94720

Director
Electronics Sciences Laboratory
University of Southern California
Los Angeles, CA 90007

Director
Microwave Research Institute
Polytechnic Institute of New York
333 Jay Street
Brooklyn, NY 11201

Director
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA 02139

Director
Stanford Electronics Laboratory
Stanford University
Stanford, CA 94305

Stanford Ginzton Laboratory
Stanford University
Stanford, CA 94305

Officer in Charge
Carderock Laboratory
Code 18 - G. H. Gleissner
David Taylor Naval Ship Research
and Development Center
Bethesda, MD 20084

Dr. Roy F. Potter
3868 Talbot Street
San Diego, CA 92106